Current state of the art in TDDFT code

Petr Koval, Olivier Coulaud, Dietrich Foerster

Current state of the art in TDDFT code

Peter Koval, Olivier Coulaud, Dietrich Foerster

Biarritz 07/10/2009

Spectroscopy versus linear response theory

- Spectroscopic properties of molecules can be explained with linear response theory
- Linear response theory: all observables can be computed with ground state wave function
- Key quantity is linear density response:

$$\delta n(\boldsymbol{r}, t) = \int \chi(\boldsymbol{r}, \boldsymbol{r}', t' - t) \delta V(\boldsymbol{r}', t') \, \mathrm{d}^3 r' \mathrm{d}t',$$

In dipole approximation, the polarizability reads

$$P_{ik}(\omega) = \int \boldsymbol{r}_i \chi(\boldsymbol{r}, \boldsymbol{r}', \omega) \boldsymbol{r}'_k \, \mathrm{d}^3 r \mathrm{d}^3 r'.$$

TDDFT linear response theory

If $$(T+V)\,\psi_E = E\psi_E$$

then $$\chi = \frac{\delta n}{\delta V} = \sum_{E\cdot F<0} (f_E - f_F)\frac{\psi_E(\boldsymbol{r})\psi_F(\boldsymbol{r})\psi_F(\boldsymbol{r}')\psi_E(\boldsymbol{r}')}{\omega - (E-F) + \mathrm{i}\varepsilon}$$

Erwin Schrödinger

$V = V_{\mathrm{ext}}$

Walter Kohn

$V = V_{\mathrm{eff}} = V_{\mathrm{ext}} + V_{\mathrm{Hxc}}$

TDDFT linear response theory

Kohn-Sham response χ_0 and interacting response χ are connected[1]

$$\chi_0 = \frac{\delta n}{\delta V_{\text{eff}}} \qquad \chi = \frac{\delta n}{\delta V_{\text{ext}}}$$

$$V_{\text{eff}} = V_{\text{ext}} + V_{\text{Hxc}} \quad \Rightarrow \quad \frac{\delta V_{\text{eff}}}{\delta n} = \frac{\delta V_{\text{ext}}}{\delta n} + \frac{\delta V_{\text{Hxc}}}{\delta n} \Rightarrow$$

$$\chi_0^{-1} = \chi^{-1} + \Sigma \Rightarrow$$

$$\chi = \frac{1}{1 - \chi_0 \Sigma} \chi_0.$$

The problem is to effectively compute the interacting polarizability

$$P_{ik} = \langle \boldsymbol{r}_i | \chi | \boldsymbol{r}_k \rangle.$$

[1] Petersilka, Gossmann and Gross, Phys. Rev. Lett., **76** (1996) 1212

Suitable basis for non interacting response

Kohn-Sham density response reads

$$\chi = \frac{\delta n}{\delta V} = \sum_{E \cdot F < 0} (f_E - f_F) \frac{\psi_E(\boldsymbol{r})\psi_F(\boldsymbol{r})\psi_F(\boldsymbol{r}')\psi_E(\boldsymbol{r}')}{\omega - (E - F) + \mathrm{i}\varepsilon}$$

Note: formula contains **products of eigenstates** $\psi_E(\boldsymbol{r})\psi_F(\boldsymbol{r})$

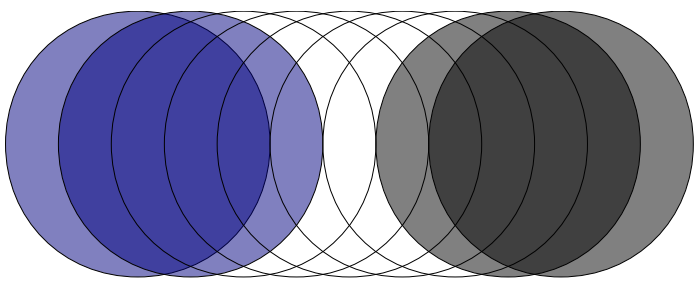
Using method of LCAO (linear combination of atomic orbitals), we get

$$\psi_E(\boldsymbol{r}) = \sum_a X_a^E f^a(\boldsymbol{r}) \quad \Rightarrow \quad \psi_E(\boldsymbol{r})\psi_F(\boldsymbol{r}) = \sum_{ab} X_a^E X_b^F \; f^a(\boldsymbol{r}) f^b(\boldsymbol{r})$$

There are

$O(N)$ localized products $f^a(\boldsymbol{r})f^b(\boldsymbol{r})$

$O(N^2)$ products of eigenstates $\psi_E(\boldsymbol{r})\psi_F(\boldsymbol{r})$



Dominant products[2]

Set of localized products $f^a(\boldsymbol{r})f^b(\boldsymbol{r})$ linearly dependent $\Rightarrow$ redundant

- find most important linear combinations of original products
- use these linear combinations as a basis

Compute a Gram matrix or metric $g^{ab,cd} = \int f^a(\boldsymbol{r})f^b(\boldsymbol{r})f^c(\boldsymbol{r})f^d(\boldsymbol{r})\, d^3r$

Diagonalize metric $gX^\lambda = \lambda X^\lambda$

Form orthogonal products $F^\lambda(\boldsymbol{r}) = \sum_{ab} X^\lambda_{ab} f^a(\boldsymbol{r})f^b(\boldsymbol{r})$

Find **vertex** $V = X^{-1} = X^T$

$$f^a(\boldsymbol{r})f^b(\boldsymbol{r}) = \sum_\lambda V^{ab}_\lambda F^\lambda(\boldsymbol{r})$$

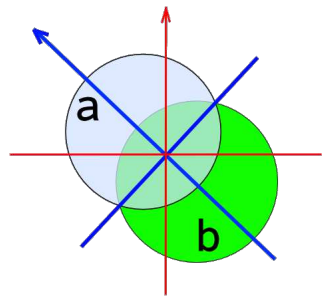
Use eigenvalue λ to define important **dominant products**

[2] D. Foerster, J. Chem. Phys. **128**, 034108 (2008)

Dominant products: constrains

- Dominant products must preserve a simple form:

$$F^{\lambda}(\boldsymbol{r}) = \sum_{j} F_{j}^{\lambda}(r) \mathrm{Y}_{jm}(\boldsymbol{R}\boldsymbol{r})$$



$\Rightarrow$ expand the dominant products about a midpoint

$\Rightarrow$ define the dominant products in a rotated frame $\boldsymbol{R}$

- Metric g must be easily diagonalizable (small):

$\Rightarrow$ define metric g within a given atomic pair

Implication: vertex becomes a sparse object

$$f^{a}(\boldsymbol{r}) f^{b}(\boldsymbol{r}) = \sum_{\lambda} V_{\lambda}^{ab} F^{\lambda}(\boldsymbol{r})$$

For a given product index λ only few V_{λ}^{ab} will be non zero.

Construction of χ_0 in $O(N^2 N_\omega)$ operations[3]

- Ansatz $f^a(\boldsymbol{r}) f^b(\boldsymbol{r}) = \sum_\mu V_\mu^{ab} F^\mu(\boldsymbol{r})$ leads to

$$\chi^0_{\mu\nu}(\omega) = \sum_{E<0,F>0} \sum_{pqrs} X_p^E X_q^F X_r^E X_s^F \frac{V_\mu^{pq} V_\nu^{rs}}{\omega - (E-F) + \mathrm{i}\varepsilon}$$

- Spectral function $a_{\mu\nu} = \mathrm{Im}\chi^0_{\mu\nu}$

$$a_{\mu\nu}(\lambda) = \sum_{E<0,F>0} \delta(\lambda - (E-F)) \sum_{pqrs} X_q^F X_s^F \; X_p^E X_r^E \; V_\mu^{pq} V_\nu^{rs}$$

- $a_{\mu\nu}(\lambda)$ via convolution

$$a_{\mu\nu}(\lambda) = \sum_F \sum_{pqrs} \rho_-^{qs}(-F) * \rho_+^{pr}(\lambda - F) V_\mu^{pq} V_\nu^{rs}$$

electronic spectral function $\rho_-^{qs}(F) = \sum_{\nu<0} \delta(\nu - F) X_q^F X_s^F$

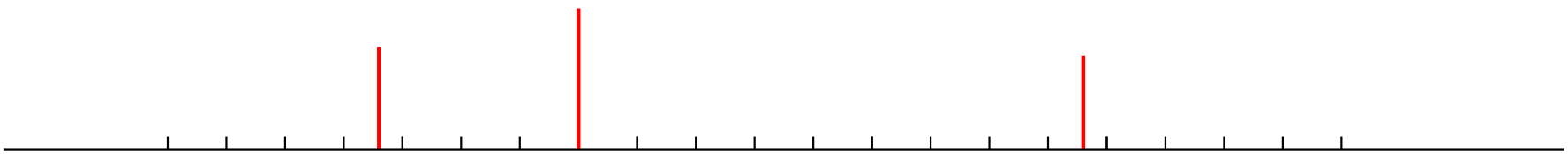
- Discretise $\lambda = E - F \Rightarrow \chi^0_{\mu\nu}(\omega)$ is again convolution

$$\chi^0_{\mu\nu}(\omega) = \sum_\lambda a_{\mu\nu}(\lambda) \cdot (\omega - \lambda + \mathrm{i}\varepsilon)^{-1}$$

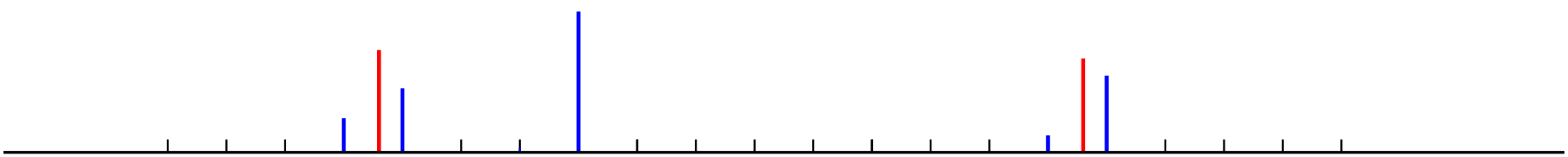
[3] D. Foerster, P. Koval, J. Chem. Phys. **131**, 044103 (2009)

Spectral function: discretisation

- FFT needs equidistant grid, but
  energy differences do not fit any equidistant grid



- Solution: “occupy” adjacent nodes



✓ Result: χ_0 exact up to discretisation

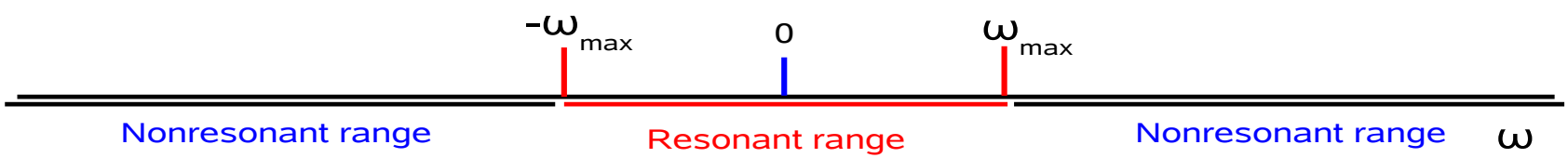
Spectral function: second window

- Spectral range of DFT eigenenergies is large: ≈ 110 eV for benzene
- Only low energy relevant $\Rightarrow$ need $\chi_0(\omega)$ in a small target window
- $a(\lambda)$ outside of target window contributes to $\chi_0(\omega)$ inside

$$\chi_0(\omega) = \int_{-\infty}^{\infty} d\lambda \frac{a(\lambda)}{\omega - \lambda}$$

- Solution: resonant and non-resonant spectral functions

$$\chi_0(\omega) = \int_{-\infty}^{-\omega_{max}} d\lambda \frac{b(\lambda)}{\omega - \lambda + i\varepsilon} + \int_{-\omega_{max}}^{+\omega_{max}} d\lambda \frac{a(\lambda)}{\omega - \lambda + i\varepsilon} + \int_{\omega_{max}}^{+\infty} d\lambda \frac{b(\lambda)}{\omega - \lambda + i\varepsilon}$$



Coulomb self energy Σ_{H}: sketch of basis

- In the basis of dominant products Σ_{H} reads

$$\Sigma_{\mathrm{H}}^{\mu\nu} = \int d\boldsymbol{r} d\boldsymbol{r}' \;\; F^{\mu}(\boldsymbol{r}) \, |\boldsymbol{r} - \boldsymbol{r}'|^{-1} \, F^{\nu}(\boldsymbol{r}').$$

  Dominant functions $F^{\mu}(\boldsymbol{r})$ are either local or bilocal.

- Generally $F^{\mu}(\boldsymbol{r}) = \sum_j F_j^{\mu}(r') S_{jm}(\boldsymbol{R}_{\mu}\boldsymbol{r}')$, $\boldsymbol{r}' = \boldsymbol{r} - \boldsymbol{C}_{\mu}$.
- Radial functions $F_j^{\mu}(r')$ are given on a logarithmic grid.

Coulomb self energy Σ_{H}: method

- $\Sigma_{\mathrm{H}}^{\mu\nu}$ can be reduced to a sum over elementary Coulomb interactions

$$\begin{aligned} E_{jm,j'm'}(\boldsymbol{c},\boldsymbol{c}') &= \int d\boldsymbol{r}d\boldsymbol{r}' \; \frac{g_{jm}(\boldsymbol{r}-\boldsymbol{c})g_{j'm'}(\boldsymbol{r}'-\boldsymbol{c}')}{|\boldsymbol{r}-\boldsymbol{r}'|} \\ g_{jm}(\boldsymbol{r}) &= g_j(r)S_{jm}(\boldsymbol{r}) \end{aligned}$$

- $E_{jm,j'm'}(\boldsymbol{c},\boldsymbol{c}') = \langle g_{jm}|p^{-2}|g_{j'm'}\rangle$ in momentum space.
- Conversion to momentum space by Talman's fast Bessel transform[4].

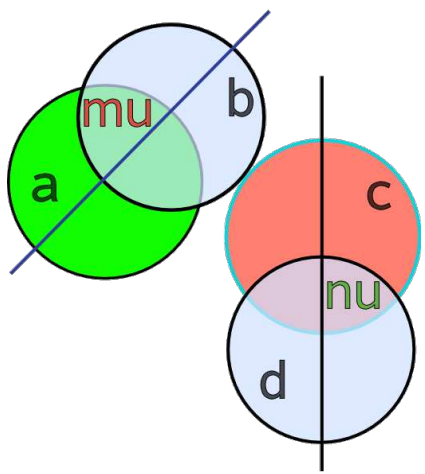
[4] Talman J. D., Comput. Phys. Commun. **180** 332 (2009)

Coulomb self energy Σ_{H}: CPU cost

- For small molecules $\Sigma_{\mathrm{H}}^{\mu\nu}$ costs $O(N^2)$
- For large molecules $\Sigma_{\mathrm{H}}^{\mu\nu}$ costs $O(N)$, because

non overlapping $F^{\mu}(\boldsymbol{r})$, $F^{\nu}(\boldsymbol{r})$

interact via their multipoles.



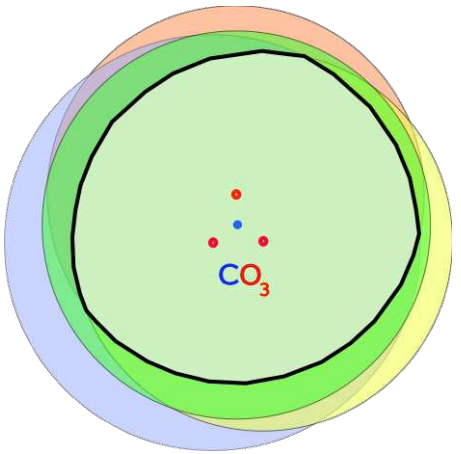
✓ No 4-center integrals involved

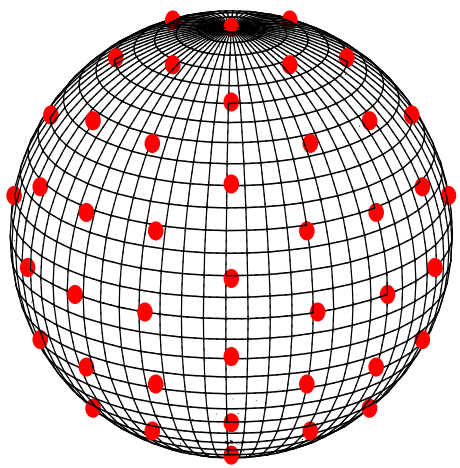
Exchange-correlation self energy Σ_{xc}

ALDA: $f_{\mathrm{xc}}(\boldsymbol{r}) \equiv \delta(t-t')\delta(\boldsymbol{r}-\boldsymbol{r}')\dfrac{dV_{\mathrm{xc}}}{dn}(\boldsymbol{r})$

$\Rightarrow \Sigma^{\mu\nu}_{\mathrm{xc}} = \int d\boldsymbol{r}\; F^{\mu}(\boldsymbol{r}) f_{\mathrm{xc}}(\boldsymbol{r}) F^{\nu}(\boldsymbol{r})$

- Spherical coordinates $\int d\boldsymbol{r} = \int r^2 dr d\Omega$
- Lebedev method for solid angle $\int d\Omega$, Gauss-Legendre for $\int r^2 dr$.
- Organize calculation in batches.





✓ Σ_{xc} costs $O(N)$.

Polarizability: CPU costs

- Non interacting response χ_0 $O(N^2 N_\omega)$
- Self energies $\Sigma_{\rm H}$, $\Sigma_{\rm xc}$ $O(N)$
- Solving $\chi^{-1} = \chi_0^{-1} - \Sigma$ with direct methods $O(N^3 N_\omega)$

☺ Fortunately $P^{ik} = d^i \chi d^k$ can be calculated iteratively

- Thanks to iterative methods $O(N^2 N_{\rm iter} N_\omega)$

✓ Interacting polarizability P^{ik}, total CPU $O(N^2 N_\omega)$

Polarizability: method

$$P^{ik} = d^i \frac{1}{1-\chi_0\Sigma}\chi_0 d^k \Rightarrow \text{avoid } O(N^3) \text{ inversion.}$$

- Bi-orthogonal Lanczos method for non-hermitian matrices
  1. $A = 1 - \chi_0\Sigma \;\rightarrow$ tridiagonal form t_{nm} $\qquad A = |n\rangle t_{nm} \langle m|$
  2. A^{-1} is easy because t_{nm} is tridiagonal $\qquad A^{-1} = |n\rangle t^{-1}_{nm} \langle m|$
  3. Interacting polarizability reads $\qquad P = P_0 t^{-1}_{11}$
- GMRES allows to control precision during iterations
  1. Solve by GMRES $\qquad (1-\chi_0\Sigma)x = \chi_0 d_i$
  2. Compute polarizability $\qquad P^{ii} = \langle d_i | x \rangle$
  3. Mean polarizability is trace of P^{ii}.

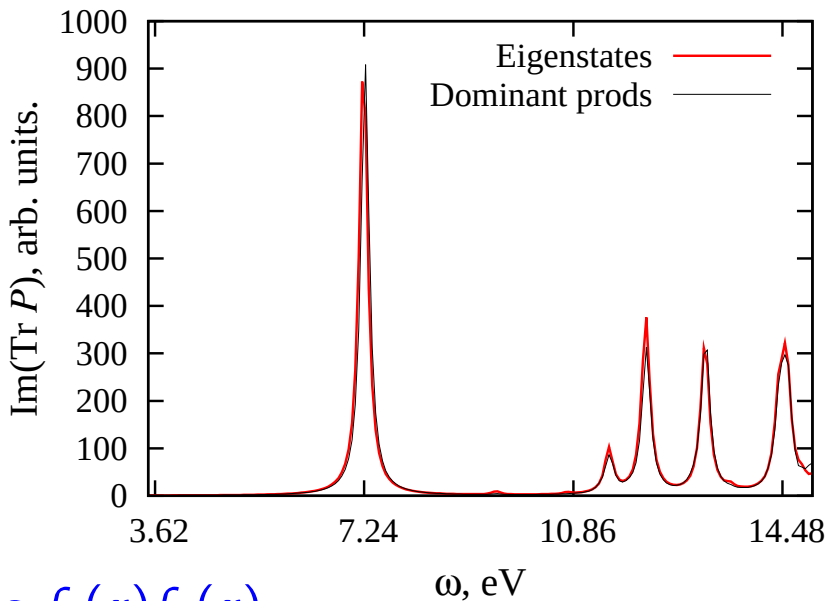
Dominant products basis: compression

How many dominant products $F^{\mu}(\boldsymbol{r})$ needed?

- Compare with exact result in the conventional basis.
- $\chi^{0}_{EF,PQ}$ is diagonal ☺, but interaction kernel costs $O(N^3)$ ☹

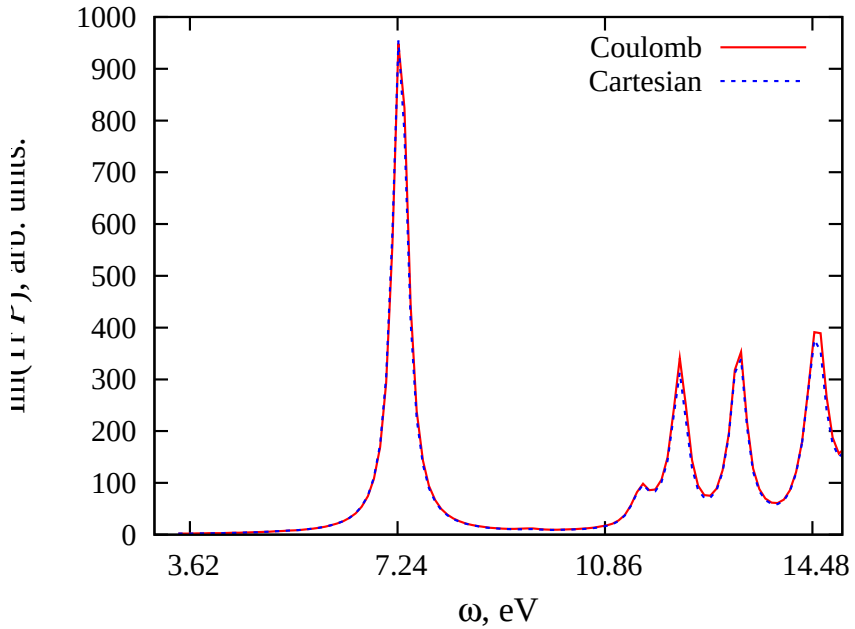
interaction kernel K via self energy Σ, vertex V and density matrix D:

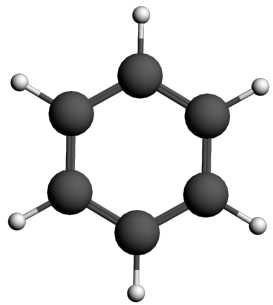
$K = (DV)\Sigma(V^T D)$



- Original basis 5832 products $f_a(\boldsymbol{r})f_b(\boldsymbol{r})$
  Dominant basis 1236 products $F^{\mu}(\boldsymbol{r})$.

Dominant products basis: Coulomb versus Cartesian metric

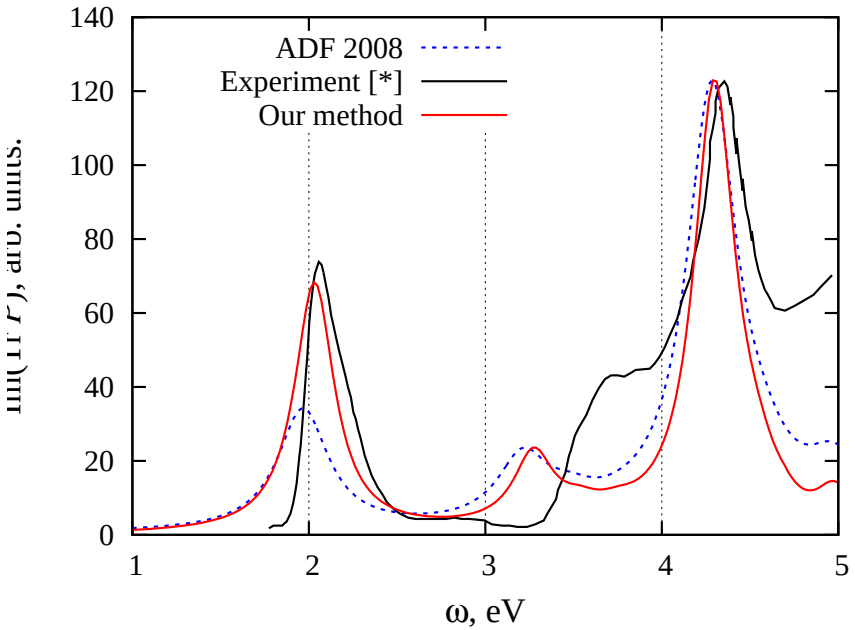


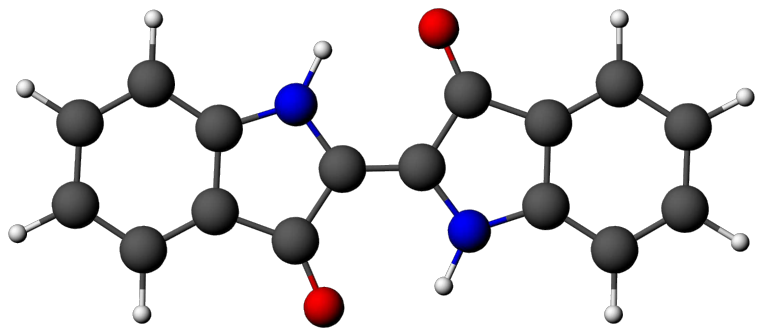


Metric	Number of products	Run time[5], s	Expression for metric
Cartesian	1158	1207	$f^a(\boldsymbol{r})f^b(\boldsymbol{r})f^c(\boldsymbol{r})f^d(\boldsymbol{r})$
Coulomb	921	816.8	$f^a(\boldsymbol{r})f^b(\boldsymbol{r})f^c(\boldsymbol{r}')f^d(\boldsymbol{r}')/\lvert\boldsymbol{r}-\boldsymbol{r}'\rvert$

[5](Dell laptop, one thread)

Dominant products basis: indigo





Program	TDDFT_LR	ADF
Wall time[6], hours	3.0	17

[*] Ross Brown, IPREM unpublished (2008)

[6] ADF with one thread on Dell laptop; TDDFT_LR with one thread on Grid5000.

Implementation: a bird's eye overview

- Input data from DFT calculation $H, S, f^a(\boldsymbol{r})$
- Diagonalize H, check S X_a^E, E
- Construct product basis $f^a(\boldsymbol{r}) f^b(\boldsymbol{r}) = V_\mu^{ab} F^\mu(\boldsymbol{r})$
- Compute response matrix $\chi_0(\boldsymbol{r}, \boldsymbol{r}', \omega) = F^\mu(\boldsymbol{r}) \chi^0_{\mu\nu}(\omega) F^\nu(\boldsymbol{r}')$
- ... self-energies $\Sigma^{\mu\nu} = \iint d\boldsymbol{r} d\boldsymbol{r}' F^\mu(\boldsymbol{r}) \Sigma(\boldsymbol{r}, \boldsymbol{r}') F^\nu(\boldsymbol{r}')$
- ... electronic excitation spectrum $P^{ik} = \boldsymbol{d}_i (1 - \chi_0 \Sigma)^{-1} \chi_0 \boldsymbol{d}_k$

Implementation: a bird's eye overview

- Input data from DFT calculation $H, S, f^a(\boldsymbol{r})$
- Diagonalize H, check S X_a^E, E
- Construct product basis $f^a(\boldsymbol{r}) f^b(\boldsymbol{r}) = V_\mu^{ab} F^\mu(\boldsymbol{r})$
- Compute response matrix $\chi_0(\boldsymbol{r}, \boldsymbol{r}', \omega) = F^\mu(\boldsymbol{r}) \chi^0_{\mu\nu}(\omega) F^\nu(\boldsymbol{r}')$
- ...self-energies $\Sigma^{\mu\nu} = \iint d\boldsymbol{r} d\boldsymbol{r}' F^\mu(\boldsymbol{r}) \Sigma(\boldsymbol{r}, \boldsymbol{r}') F^\nu(\boldsymbol{r}')$
- ...electronic excitation spectrum $P^{ik} = \boldsymbol{d}_i (1 - \chi_0 \Sigma)^{-1} \chi_0 \boldsymbol{d}_k$

Points to mention for each step of the program

- Sketch the algorithm and implementation
- Type of parallelization chosen/implemented
- Improvements done (since June 2009)
- Wall time and speedup factors on Grid5000 / M3PEC / Dell Laptop

Construct product basis: algorithm

Algorithm 1: Generation of bilocal vertex

1 **forall** *atom pairs* **do**

2 Compute an expansion of $f_a(\boldsymbol{r})f_b(\boldsymbol{r}) = \sum_j F_j(r)Y_{jm}(\boldsymbol{r})$ (Talman)

3 Convert the expansion to momentum space $F_j(r) \rightarrow F_j(p)$ (Talman)

4 Build Coulomb metric g using $F_j(p)$

5 Diagonalize the metric $gX^\lambda = \lambda X^\lambda$

6 Build dominant products and vertex with $\lambda = \lambda/10$

7 Allocate intermediate storage for dominant products and vertex

8 Store the generated data

9

Construct product basis: parallelization

- Local vertex is a fast operation – no reason to parallelize
- Generation of bilocal vertices takes few seconds ($\sim$ 10 s)
- Vertices and products are needed on all nodes

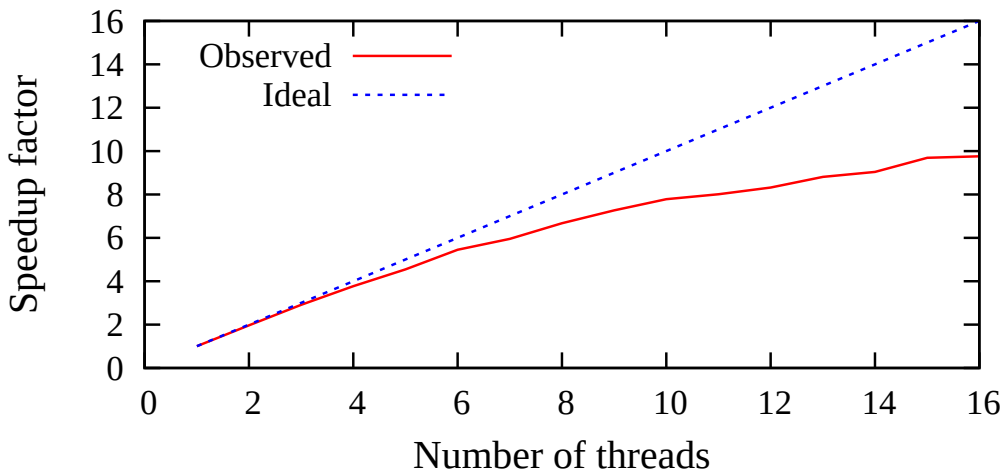
✓ Bilocal vertices to be parallelized with OpenMP: done

Construct product basis: improvements

1. Intermediate storage introduced: storage requirements grows $O(N)$
2. Cartesian metric is implemented for testing
3. Loop over atom pairs is rewritten
4. Rotation matrices are initialized once
5. Talman's fast Bessel transform is reshaped in FFTW fashion
6. Loop over atom pairs is parallelized (OpenMP)

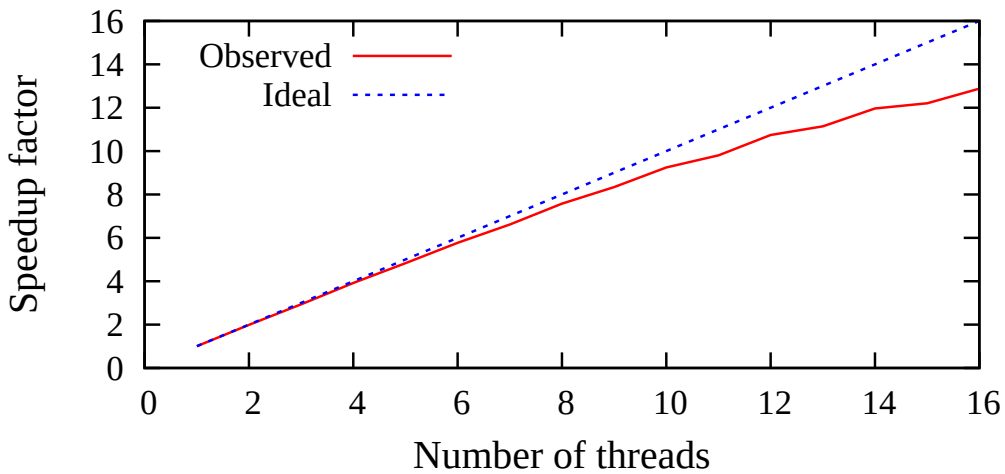
Construct product basis: timing with benzene

OMP_NUM_THREADS	Dell laptop	Grid5000	M3PEC
Sequential	4.58	4.30	7.99
1	8.60	4.51	9.01
2	2.60	2.24	4.58
8	-	0.913	1.35
16	-	-	0.924
Max speedup	3.30	4.94	9.75



Construct product basis: timing with indigo

OMP_NUM_THREADS	Grid5000	M3PEC
Sequential	20.7	38.8
1	21.3	42.3
2	10.8	21.3
8	2.88	5.58
16	-	3.28
Max speedup	7.40	12.9



Compute response matrix: algorithm

Algorithm 2: Calculation of response matrix

1 Generate Fourier transform of density matrices $\rho^{\pm}(\tau)$

2 **forall** *atom quadruplets* **do**

3 1. Multiply $\rho^{\pm}(\tau)$ with vertices $V\_rho = V\rho^{\pm}(\tau)$

4 2. Find Fourier transform of spectral function $a(\tau) = V\_rho \cdot V\_rho$

5 Do steps 1 and 2 for $\omega_{\max 2}$ – compute spectral function $b(\tau)$

6 Nullify $b(\tau)$ in first (target) window

7 Prepare second convolution $\chi^0(\tau) = a(\tau) * FFT(\omega - \lambda + \mathrm{i}\varepsilon)$

8 Fourier transform of response $\chi^0(\omega) = FFT(\chi^0(tau))$

9 Interpolation of second window response $\chi_2^0(\omega)$, update $\chi^0(\omega)$

10 Store the generated data

11

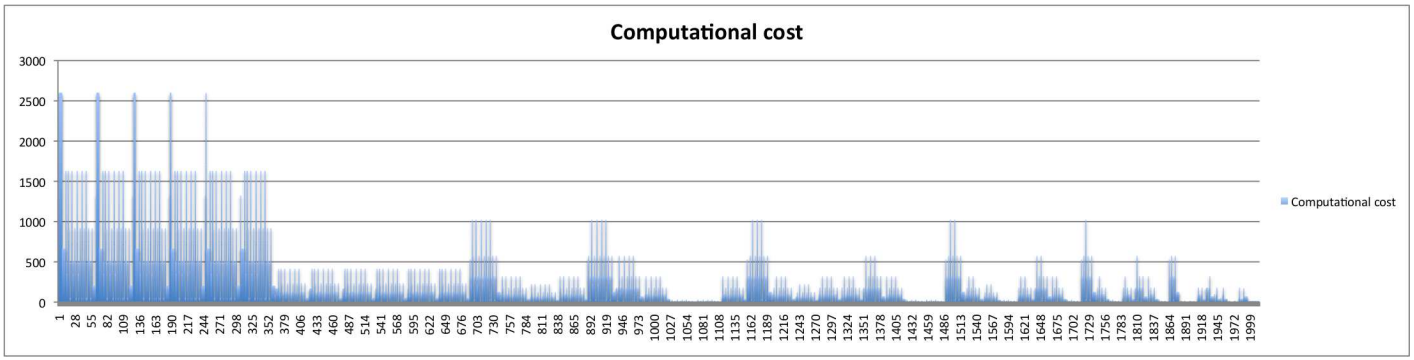
Compute response matrix: parallelization

- Response matrix takes much memory (currently 11 GB for indigo)
- Although $O(N^2 N_\omega)$, calculation is long (3 hours for indigo)
- Calculation of matrix elements is independent task
- Available machines contain few CPU on one node (16 on M3PEC)
- Available machines consists of many nodes (hundreds)

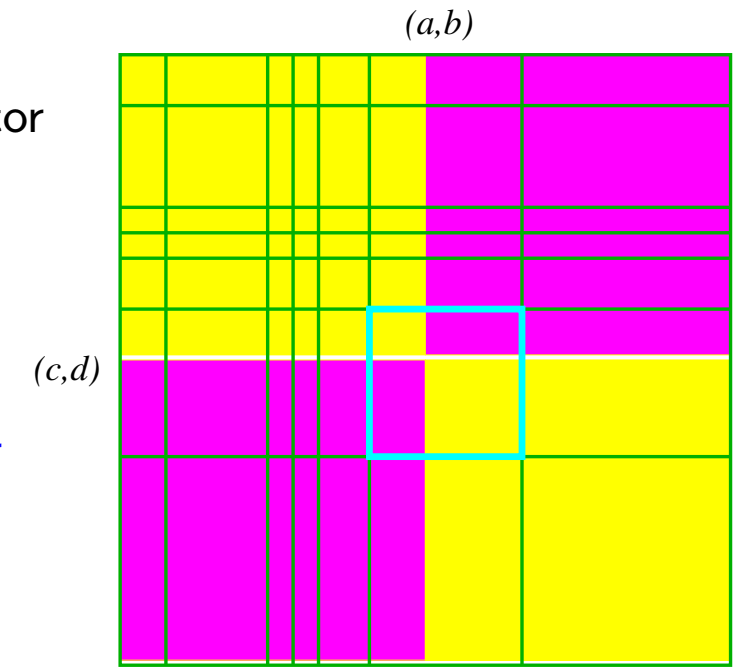
- Response matrices to be parallelized with OpenMP and MPI

Compute response matrix: parallelization

- Amount of work **is different** for atom quadruplets: load balancing



- Reorder quadruplets to obtain a good schedule for OpenMP
  - Iterative method requires matrix-vector products
  - There are MPI parallelized BLACS implementations: use them
  - Give up the "natural" blocks in favor of block-cyclic decomposition



Compute response matrix: OpenMP parallelization

Algorithm 3: OpenMP parallelization of quadruplet loop

```
1 !$OMP PARALLEL
2 allocate private and threadprivate variables here
3 !$OMP DO SCHEDULE(DYNAMIC,10)
4 forall atom quadruplets do
5 |   Compute block of response matrix
6 |   Store the generated data
7 !$OMP END DO
8 deallocate private and threadprivate variables here
9 !$OMP END PARALLEL
```

✓ OpenMP parallelization is done.

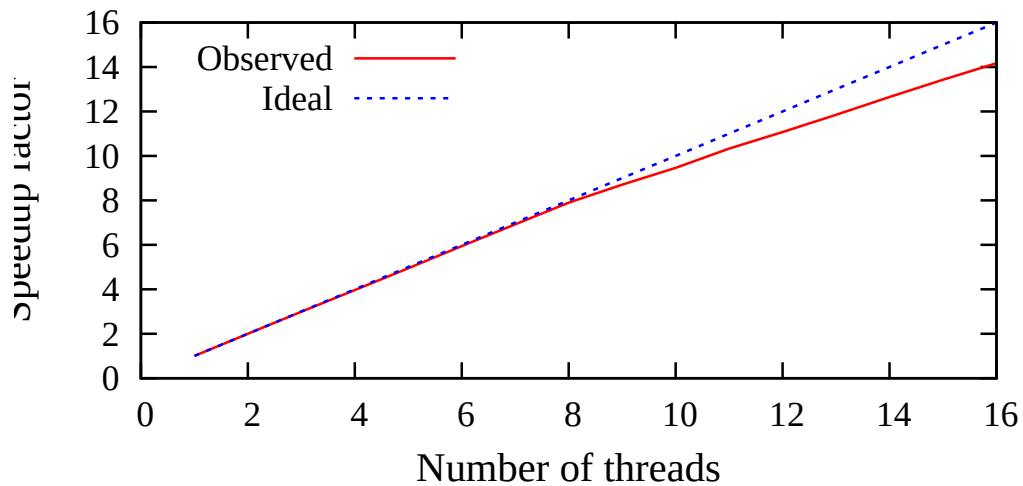
Compute response matrix: improvements

- Sparse data storage for vertex (less memory & more speed)
- Quadruplets of atoms introduced (half of second convolutions ✗)
- Result is saved in single precision (halves memory requirement)
- Skip unnecessary vertex–matrix multiplications in diagonal blocks ✗
- Automatic arrays removed✗
- Storage of intermediate vertex–matrix products shaped
- Single precision in vertex–matrix products
- OpenMP parallelization of the main loop
- Threadprivate variables for temporary arrays

✓ At least factor 6 speedup. OpenMP scaling on Grid5000 improved.

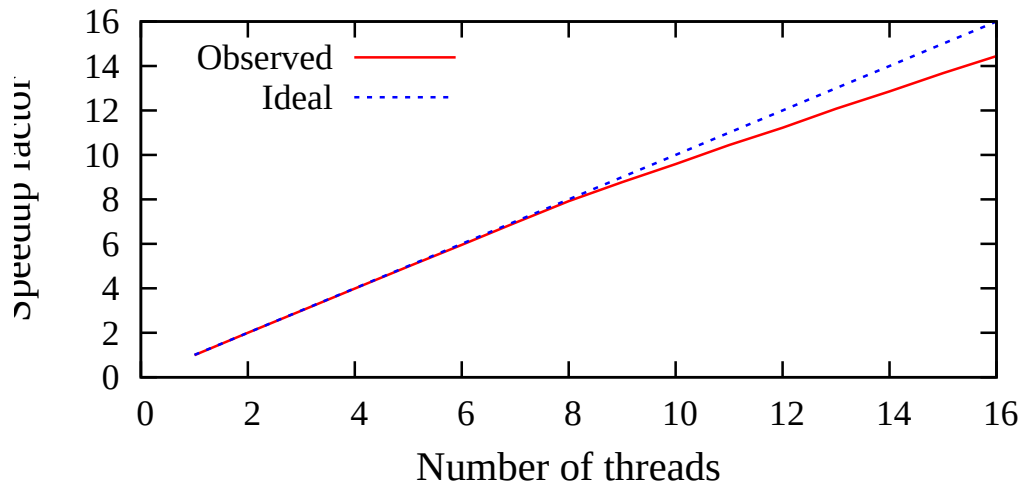
Compute response matrix: timing with benzene

OMP_NUM_THREADS	Dell laptop	Grid5000	M3PEC
Sequential	709	725	752
1	646	734	751
2	625	375	376
8	-	108	95.0
16	-	-	53.0
Max speedup	1.03	6.79	14.2



Compute response matrix: timing with indigo

OMP_NUM_THREADS	Grid5000	M3PEC
Sequential	9351	10701
1	9664	10697
2	4840	5346
8	1479	1351
16	-	741
Max speedup	6.32	14.4



Coulomb interaction matrix: algorithm

Algorithm 4: Calculation of Coulomb interaction matrix

1 Compute fast Bessel transform of all radial functions $F_l^\lambda(p)$
2 Compute Wigner rotation matrices
3 **forall** *atom quadruplets* **do**
4 | Integrate products of radial functions with $j_l(pR)/p^2$
5 | Rotate contributions to Coulomb integral
6 | Sum up the contributions from different angular momentum
7 | Store the generated data
8

Interaction matrices: parallelization

- Interaction matrices take less memory
- Calculation of matrix elements is independent task
- Parallelization strategy depends on speed of communication during last step (iterative construction of polarizability)

- Response matrices to be parallelized with OpenMP and possibly MPI

✓ OpenMP parallelization is done.

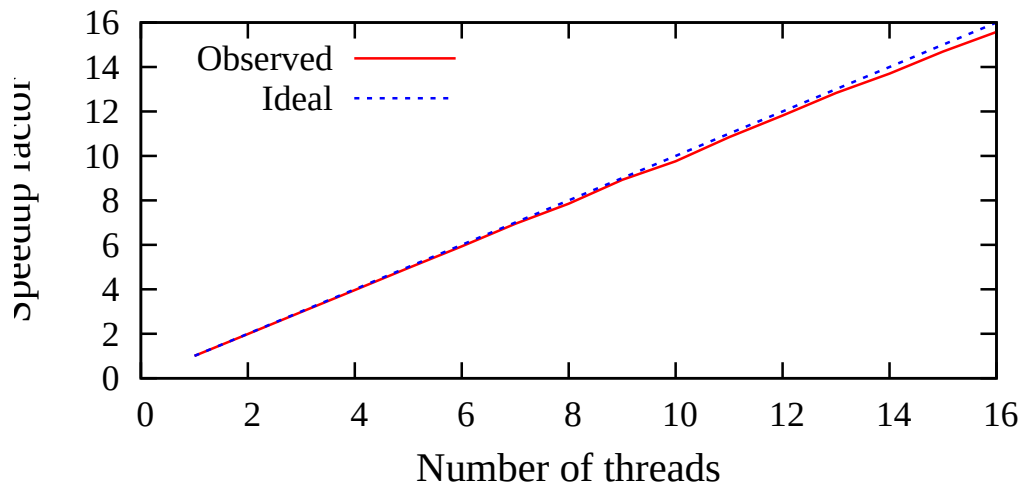
Interaction matrices: improvements

- Quadruplets of atoms introduced
- OpenMP parallelization of the main loop
- Automatic arrays removed✗
- Threadprivate variables for temporary arrays
- Indexing array is remove ✗

✓ Some speedup. Good scaling on Grid5000.

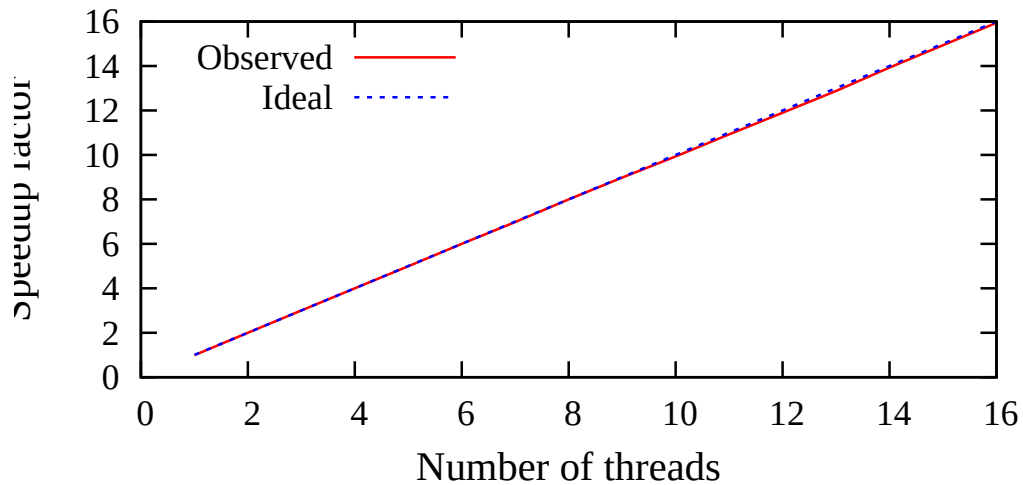
Coulomb interaction: timing with benzene

OMP_NUM_THREADS	Dell laptop	Grid5000	M3PEC
Sequential	42.7	38.2	78.6
1	42.5	37.4	210
2	21.4	18.6	105
8	-	4.71	26.7
16	-	-	13.5
Max speedup	1.98	7.93	15.6



Coulomb interaction: timing with indigo

OMP_NUM_THREADS	Grid5000	M3PEC
Sequential	393	757
1	365	2765
2	181	1380
8	45.6	346
16	-	174
Max speedup	8.04	15.9



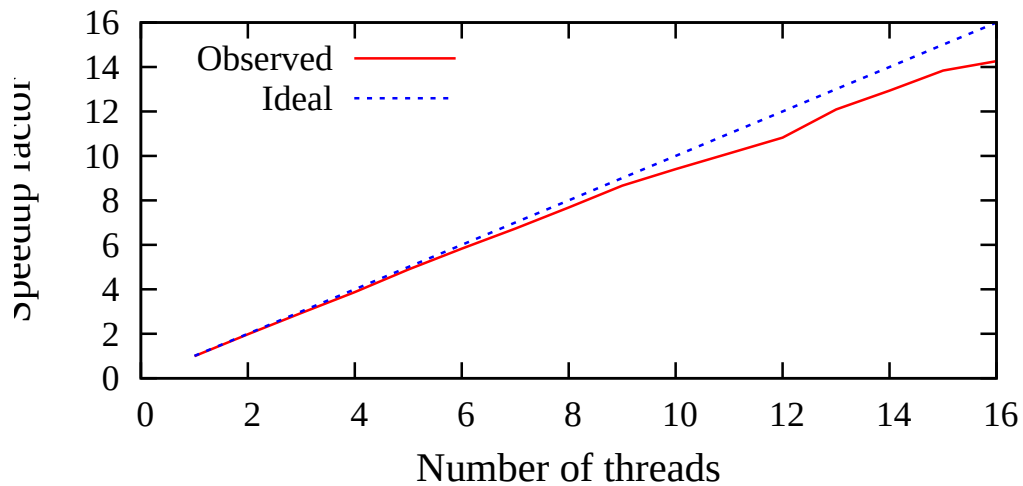
Exchange–correlation matrix: algorithm

Algorithm 5: Calculation of exchange–correlation interaction matrix

```
1  Precompute rotation matrices for fast computation of F^λ(r)
2  forall atom quadruplets do
3      Initialize batch variables; Nullify the result;
4      forall Gauss–Legendre points ∫ dr do
5          forall Lebedev points ∫ dΩ do
6              r = (r, Ω)
7              Compute values of all dominant functions
8              Compute value of Σ_x(r) and Σ_c(r)
9              Update corresponding integral
10
11     Store the generated data
12
```

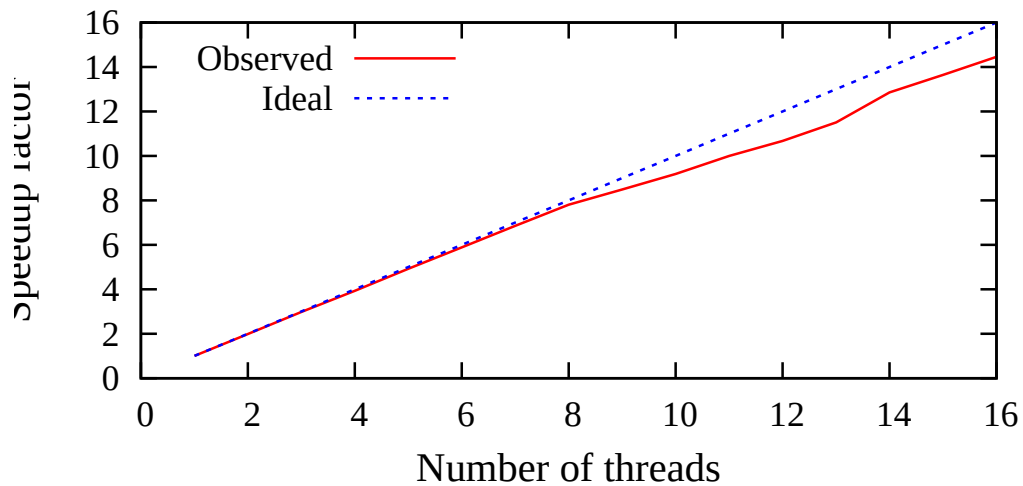
Exchange–correlation interaction: timing with benzene

OMP_NUM_THREADS	Dell laptop	Grid5000	M3PEC
Sequential	79.29	97.22	141.2
1	80.45	99.20	190.4
2	41.86	49.61	96.37
8	-	13.12	24.81
16	-	-	13.34
Max speedup	1.92	7.56	14.27



Exchange–correlation interaction: timing with indigo

OMP_NUM_THREADS	Grid5000	M3PEC
Sequential	365	524
1	368	583
2	186	293
8	50.9	74.7
16	-	40.3
Max speedup	7.23	13.0



Sequential remnants in the code: indigo

Subprogram	Grid5000, 8 threads	M3PEC, 16 threads
V_{μ}^{ab}	2.88	3.28
$\chi^{0}_{\mu\nu}$	1479	741
Σ_{H}	45.6	174
Σ_{xc}	50.9	40.3
P_{ik}	755	686
Total time	2345	1677
Remnants	11.6	32.4

Conclusions

- ▶ Fast algorithm allows for a $O(N^2 N_\omega)$ scaling in true polarizability
- ✓ Most of improvements have been done in response
- ✓ Current implementation is faster than ADF in indigo molecule
- ✓ Most of the program is parallelized with OpenMP
- 🕒 MPI parallelization remains to do

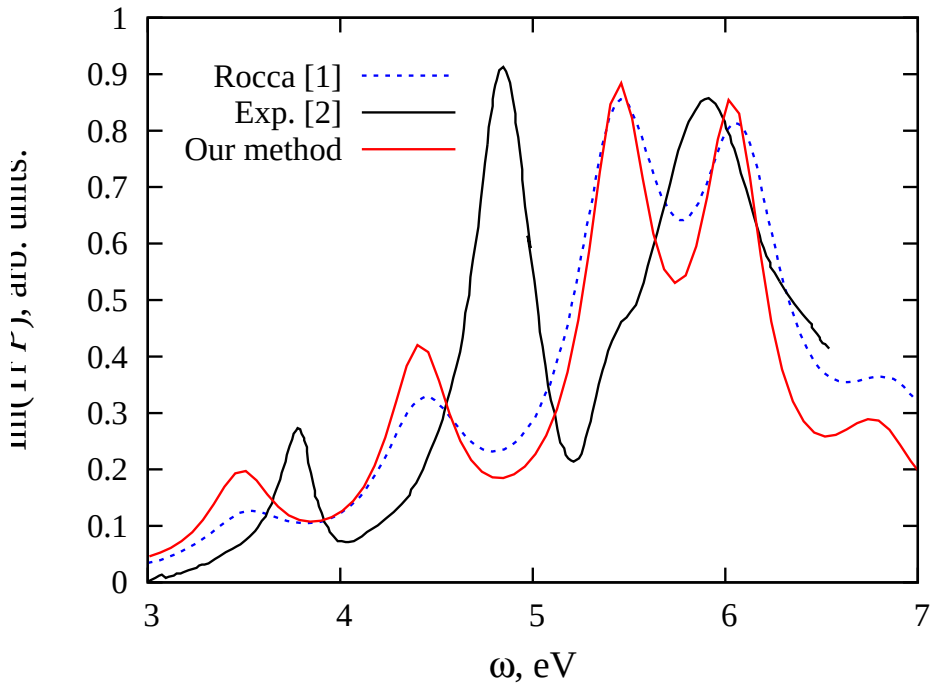
Acknowledgments

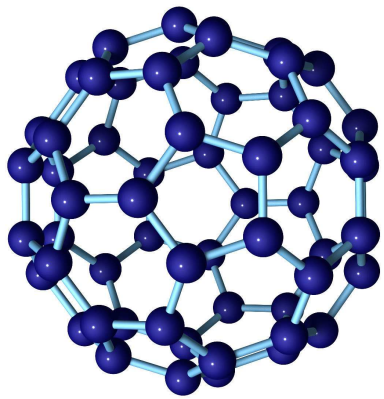
- James Talman, London, Canada
- Mark Casida, Grenoble, France
- Ross Brown, Pau, France

$$ Work supported by ANR-project NOSSI

Nouveaux Outils pour la Simulation des Solides et des Interfaces

Largest molecule computed with our method – C_{60}





- Number of dominant products 7800
- Memory (7800*7800/2*8*(128-53)/1024/1024/1024) 17 GB
- Walltime (Grid5000, 8 threads, 128 freq. points, for χ_0) 7193 s

---

[1] Dario Rocca, PhD Thesis (2007)

[2] Bauernschmitt *et al*, Am. Chem. Soc. 120, 5052 (1998) (in hexane).